IIT/K COMPUTER NETWORK—IBM-1800 HARDWARE INTERFACE

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

By

C. V KRISHNA

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

JULY, 1979

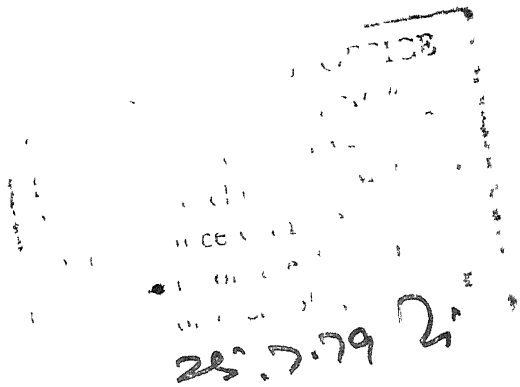
CERTIFICATE

This is to certify that the work on the 'IIT/K COMPUTER NETWORK - IBM 1800 HARDWARE INTERFACE' is done by Sri C.V. Krishna under our supervision and has not been submitted elsewhere for a degree.

A.S. Sethi
Lecturer
Computer Science Program
Indian Institute of Technology
Kanpur 208 016

R.M.K. Sinha
Assistant Professor of
Electrical Engineering, and
Computer Science
Indian Institute of Technology
Kanpur 208 016

Kanpur
July 1979



ACKNOWLEDGEMENT

I am grateful to Dr. A.S. Sethi and Dr. R.M.K. Sinha my thesis supervisors for their constant help and guidance throughout this project. My sincere thanks to Dr. A.S. Sethi who inspired us to take up this project and made this successful.

Thanks are due to Sanjeev Kumar and R.K. Jain who helped me in testing the hardware interface.

I would like to thank Sri H.K. Nathani for his excellent typing and Sri H.S. Tripathi for neat cyclo-styling.

My thanks are also due to my friends and colleagues especially Shaunak R. Pawagi but for whom my stay here would not have been so pleasant and memorable.

Finally, I wish to express my appreciations for the cooperation I got throughout this project from my friends and co-workers Sri N.S. Narayanan and Sri Debasis Das.

- C.V. Krishna

Kanpur
July 16, 1979

ABSTRACT

A system is proposed to link up DEC-1090, TDC-316 and IBM 1800 at the IIT Kanpur Computer Centre. These three machines are connected in a star configuration which uses a microcomputer system (ECIL's MICRO-78) as the central switch. The connections are full duplex with a data rate of 10K bauds. The network is designed for resource sharing and for experimental work in various aspects of computer networks. Flexibility is the prime concern in the design of the system. This report describes the overall system design, line protocol design and the hardware implementation of the line protocols on the IBM 1800 side.

-

CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

A computer network is a complex collection of many types of resources, including data bases, programs, operating systems and special purpose hardware all of which are capable of being accessed from any other resource in the network [KAHN 72]. A computer network consists of two or more computers interconnected in such a way as to achieve the required goal in an efficient manner. A principle motive in computer network development is to provide a convenient and economic method for sharing a wide variety of resources. Such a network which provides sharing of resources is called a 'Resource Sharing Network'. Another application of computer networks is to share the load of a computer among many other computers connected in the network. There are also many special purpose networks used in air-line reservation systems, banking, stock exchanging and public service applications.

The ARPA (Advanced Research Projects Agency) Net in the U.S.A. is one of the most advanced examples of a computer communications network [KLEI]. ARPA network came into existence in the late 60's for the sharing of resources among a large number of computer centres. Other examples of networks are SITA for international

airlines reservations [BRAN 72], NASDAQ for automated quotations of securities [FRAN 72], NPL in England for experiments in data communications [DAVI 68] and ALOHA in Hawaii which uses radio channels for communication [ABRA and KUO].

1-2. BASIC CONCEPTS

A computer network may be viewed as a set of nodes connected together by edges (channels). Information is transferred in a network from one node to another in units called messages. The path followed by a message from the source node to the destination node is determined as it passes from node to node through the network; no continuous link is established from the source node to the destination. This form of information transfer is called message switching and contrasts with circuit switching used in telephone networks where a physical path is set up in advance of the message transfer. Some networks break up a large message into smaller units called packets. These packets are then sent individually through the network and re-assembled at the destination to get the original message. This is called packet switching [CROW 75]. To carry out the various tasks associated with information transfer in a message-switched or packet-switched network, the network usually employs additional computing power in the form of communications processors which are called

interface processors [ORNS 72]. These processors together with the communication lines are called the 'communications subnetwork'. This subnetwork is more or less transparent to the users of the network. The interface processors form the interface between the subnetwork and the main computers called 'Hosts' [WALD 72].

The nodes of a computer network may be interconnected in various ways. Some of the important configurations are star, loop, fully connected, tree, distributed and hierarchical. The configuration chosen for a network has a significant effect on the design of the network and the way it functions. For local networks covering a small geographical region, the star, loop and distributed configurations are fairly commonly employed [KAHN 72].

For proper communication between two processors, some handshake procedures are necessary to govern the exchange of data. These handshake procedures are called 'protocols' [FRAS 76]. When two processors face each other across a communication line, a protocol is the set of their agreements on the format and relative timings of messages to be exchanged. When users at different locations in the network wish to exchange information, the message must go through many levels of protocols to

reach its destination. This is because protocols are usually arranged in a hierarchy to facilitate implementation and understanding. Each level of protocols in this hierarchy has a specific function to perform. The lower level of protocols are concerned with details of communication while higher levels are function-oriented [FRAS 76]. The lowest level of protocols are called 'line protocols' and describe mainly the communication between two processors connected by a physical line.

The main functions of line protocols are to maintain synchronisation between sending station and receiving station, initiating message transfer, terminating message transfer, acknowledgement mechanisms to inform source of the correct receipt of messages, error detection and recovery techniques and time-outs [GRAY 72, STUT 72]. These are explained in detail in third chapter of this thesis.

1-3. IITK COMPUTER NETWORK

Although computer networks is a fast developing field, not much work has been done in this area in our country. TIFR-Bombay has developed an experimental network which links TDC-316 machines to a DEC-1090 by telephone lines. Air-India is also involved in a big centralized computer network for airline reservations.

The central switching computer is a UNIVAC machine located in Bombay and will be connected to many minicomputers located all over India by telephone lines. There is a proposal from Indian Space Research Organisation (ISRO) to link up its different computers located in Ahmedabad, Trivandrum, Bangalore and Sri Harikota by radio channels.

In this background, the Computer Centre at IIT-Kanpur has taken up a project involving the construction of a small local computer network. The network is both experimental and utilitarian. It will provide the needed practical experience in the design and implementation of a network and also serve as a vehicle for experimental studies in network flow control, routing, protocols and performance evaluation. However, the network will also provide services to the users of the Computer Centre and is planned to ultimately become an essential facility of the Centre.

The Computer Centre currently has three machines available to the general user: the DEC-1090 (which has recently arrived as a replacement for the earlier IBM 7044/1401 systems), the TDC-316 and the IBM 1800. Although the DEC-1090 is the central computation facility, its utility can be greatly increased by linking all the three machines in a computer network. The DEC users would then be able to access the card punch of the

IBM 1800, a device not available on the DEC system. Moreover, the network can be used to convert old 7-track magnetic tapes to the 9-track tapes needed by DEC by using the 7-track tape drive available on the IBM 1800. The TDC-316 is currently planned to be used as a remote job entry station for the DEC system. There is also a plan to use it as a concentrator for terminals thus allowing easy system expansion.

To be able to satisfy these diverse requirements, it is necessary that the network should be flexible and not limited in any way by the design of the initial hardware. Because of resource limitations, the hardware should also be cheap and simple. Thus, simplicity, economy and flexibility became the most important goals of the network.

For reasons - to be discussed in the next chapter, a star configuration was chosen for the network and a microcomputer, MICRO-78, produced by Electronics Corporation of India Ltd., Hyderabad, was chosen as the central node of the network. This thesis describes the design and implementation of the hardware interface for IBM 1800, a part of the network. The interfaces for TDC-316 and MICRO-78, other parts of the network are described in [NSN 79] and [DDAS 79] respectively, which complement this work.

1-4. OVERVIEW OF THE THESIS

In the second chapter of this thesis is a brief description of the system is given. The network configurations and constraints on the network design are discussed in detail. The design and implementation of line protocols are discussed in Chapter 3. The communication between MICRO-78 and IBM 1800 is explained in this chapter. The hardware design and implementation details of the sending interface and the receiving interface of the IBM 1800 are elaborated in Chapter 4 and 5 respectively. The last chapter deals with further improvement of this network and the scope for developing higher level protocols for this network.

-

CHAPTER II

SYSTEM DESCRIPTION

The goals of the network and the constraints imposed on the network structure and design are discussed in this chapter. In designing this network a systematic approach has been followed. The different steps in the network design are listed below.

1. Define the aims and applications of the network.
2. Select the configuration of the network.
3. Design the line control procedures for the network.
4. Estimate the hardware complexity.
5. Go on repeating Steps 2,3,4 until the aims listed in Step 1 are more or less satisfied.
6. Design the software for the network to make effective use of the hardware interfaces.

2-1. GOALS OF THE NETWORK

As the system is going to be used in a campus environment, the main aim of the network is to provide flexibility rather than efficiency. In order to achieve a flexible network operation, the maximum control should be given to network software. Thus the hardware design should impose the minimum possible limitations on the options open to the software designer. This in turn indicates that the hardware should be simple, general and logically similar

in all the machines. This would be economical also. The system should be useful and it should be possible for the users of any machine to get access to system facilities and peripherals of the other two machines. The overall structure of the network software and hardware should be simple to understand and its hould be easy to carry out any experimental work on networks.

2-2. NETWORK CONFIGURATIONS

Although computer networks in general have a wide variety of configurations, three possible configurations are most suitable for a small local network like ours. They are star-configuration, data-loop network and fully interconnected network. A star-network has a central node to which all other nodes are directly connected. The advantages of a star configuration are simplicity and economy. A disadvantage of such a network is that it may have large line lengths if the nodes are far from the central node, but in our case, since the distances involved are small, the line lengths will not be excersive. Another disadvantage is that the system is heavily dependent on the central node, hence the reliability is quite low. Data loops are operational in a few other universities, but the main problem with data-loops is the complexity of the line control procedures and the resultant hardware complexity. Designing a flexible system using

data loops is fairly complicated and was not taken up because of our limited resources. A fully or partially connected network is not called for, since reliability is not a major consideration. The data rates needed for our proposed applications are also not large enough to justify the higher cost and complexity of such a network.

After considering all the above points, a star network was chosen with IBM 1800, TDC-316 and DEC-1090 as the three outer nodes and MICRO-78 as the switching computer. In order to explore the possibilities of a micro-computer as a switching computer, a MICRO-78 has been selected for switching purpose. The configuration is shown in Figure 2-1. Having chosen the star configuration, the choice of a full-duplex serial data transmission between any pair of nodes is almost automatic, since it is the simplest and most efficient form of communication for the distances involved.

2-3. DATA TRANSFER MODES AND DATA RATES

Since data communication is full duplex, every machine in the network is effectively connected to the MICRO-78 by a pair of simplex lines. These lines terminate at each end in a hardware interface as shown in Figure 2-2. The sending interface takes data from the machine and transmits it over the line in a suitable form. The receiving interface receives the incoming data and hands it over to the machine.

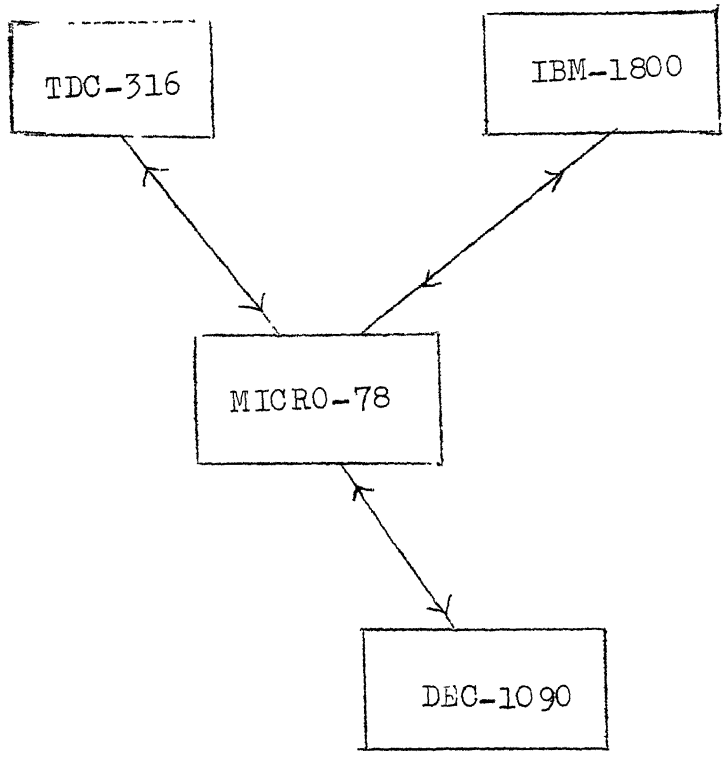


Figure 2-1: System Configuration.

SI - Sending Interface
RI - Receiving Interface

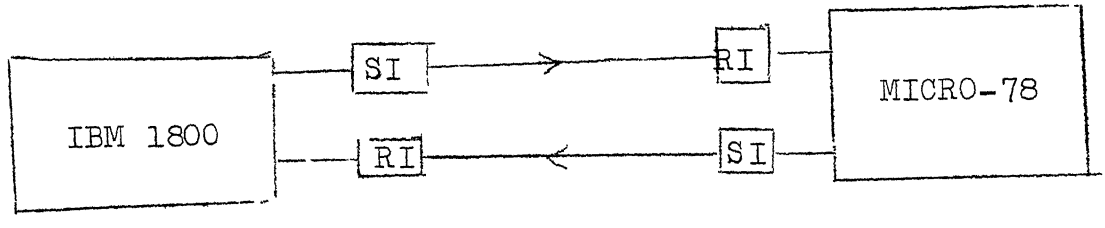


Figure 2-2: Link between the processors

Two modes of data transfer between the interface and machine are possible: (i) By interrupts, (ii) By DMA. The choice between the two depends on the ease of implementation in hardware and efficiency of transfer. In the interrupt mode, the Pc is interrupted by the interface for each byte sent or received. If the interface has a single byte buffer, the time available for servicing the interrupt is equal to the transmission of one byte. In DMA mode, the byte is obtained from or deposited into memory without the intervention of the processor. Since the overhead for DMA is much less, it can support very high data rates, typically 250 K bits/Sec out the most important point is that what the Pc does through the software in the interrupt routine, has to be done by hardware in DMA mode. DMA interface is much more complicated than interrupt interface, interms of hardware in case of TDC-316 and MICRO-78. In the case of IBM-1800, special digital input/output groups are available and data can be transmitted by a special DMA type of operation called 'data channel operation'. In confirmity with our goal of hardware simplicity, it was decided to have interrupt interfaces for TDC-316 and MICRO-78 and data channel transfer for IBM-1800.

As 9.6 k bits/sec is the standard rate of transmission in the DEC-10 synchronous interface, we decided to transmit and receive the data at 10K bits/sec for our convenience.

The transmission is bit serial and byte serial over the line. To verify that 10K bits/sec is not too much for MICRO-78 to handle, it may be seen that, assuming each byte is 8 bits + 1 parity bit and processor gets interrupted for each byte, it gets about 900μsec between each interrupt, at any interface. Assuming it has to handle 6 interfaces (3 sending and 3 receiving) at the same time, it gets about 150μsec for each interrupt service routine which is more than sufficient for executing a fair amount of software.

2-4. AN OVERVIEW OF THE COMPLETE SYSTEM

From the above discussions, the following specifications emerge for our network.

* A star configuration with TDC-316, IBM-1800 and DEC 1090 at the three nodes and a MICRO-78 as the central switch is envisaged.
* Full duplex transmission at 10K bauds chosen.
* TDC-316 and MICRO-78 are going to operate in byte interrupt mode and IBM 1800 is going to use the data channels.

The next step is to design a set of line control procedures for the communication between two processors. These are described in Chapter 3. In the remainder of this chapter, we give a brief description of the hardware structure of the IBM 1800.

2-5. BRIEF DESCRIPTION OF IBM-1800

The IBM-1800 in our Computer Centre has a memory of 16,384 words of 16 bits each with storage cycle time of 4μsec [FUN 1800]. Standard features of the system are

* three data channels
* Three index registers
* Three interval timers
* 12 levels of interrupt
* Operations monitor
* Storage protection
* Power failure protection, etc.

Our configuration has one tape unit, disk drive, paper tape punch, paper tape reader, plotter, card-reader, card punch and a typewriter. This is basically a process-control computer.

Twelve levels of interrupts are there in this system. Sixteen seperate interrupts can be assigned to each interrupt level. These interrupts are generated by programmed instruction, data processing I/O units, process I/O units and features and the attachment circuitry. The interrupt facility includes one 16-position interrupt level status word (ILSW) for each interrupt level.

The I/O devices of the 1800 system contain status indicators. The on/off condition of each status indicator informs the operating program of the status of an operation or device. Some of the status indicators may reflect a condition in the device or process that requires a program response. These indicators are assigned to interrupt level and initiate interrupt requests when they are turned on.

Status words used in the 1800 are -

* Device Status Words (DSW): A unique DSW exists for each device. Some devices have more indicators than can be contained in one word.

* Process interrupt status words (PISW): These indicators are physically located in the customer area and are turned on by closing a contact or the shifting of a voltage in a remote customer process.

* Interrupt Level Status Word (ILSW): Each ILSW bit is assigned to a PISW or a specific device. PISW indicators are ORed into the assigned ILSW bit positions. Similarly all DSW interrupt indicators from the assigned device are ORed into the single ILSW bit position assigned to that device. Whenever anyone of the ORed interrupt indicators from a PISW or DSW is active, the respective ILSW bit is turned on, causing the interrupt level to request service.

2-5.1 Data Channel Operation: Data channels provide a method of controlling data transfer between core storage and I/O devices without requiring execution of an XIO instruction to effect transfer of each data word. A data channel is initiated for a data transfer operation with a single XIO instruction. The XIO specifies the location of a data table in core storage and if necessary the number of words associated with the data transfer operation. The data channel then takes control of the data transfer operation while program execution resumes.

Some devices operate under direct program control or data channel control also. They include analog input, analog output, digital input and digital outputs.

2-5.2 Process I/O: The process I/O is divided into four general categories as analog input, analog output, digital input and digital output. Here only digital inputs and digital outputs are discussed.

(a) Digital Input: This feature enables the processor-controller to accept real time digital information in a digital format. Digital input is brought into the system in 16 bit groups. The format may be in any form.

Digital input has a 'digital input data channel adapter', which adapts digital input to a P-C data channel to enable the digital input in the cycle stealing mode of operation.

Operation with external sync: The digital input/data channel adapter provides the external synchronisation. An XIO instruction (Initialize read-synchronized) is executed to develop an external sync ready signal. When external device senses this ready signal, it must transfer the data to the addressed digital input group and then send an external sync signal to the digital input data channel adapter. The external sync. signal initiates a core storage cycle to read the data into a core storage word and turns off external sync. ready. The digital input feature will be interlocked under external sync. until word count equals to zero for last external sync is obtained. However, the P-C can execute instructions while digital input feature is interlocked. The various types of digital inputs are digital

input (contact), digital input (voltage sense), process interrupts (contact) and process interrupts (voltage sense).

(b) Digital Output: Digital outputs in the 1800 are used in groups of 16 bits. Only two types of digital outputs are available: electronic contact operate (ECO) and register outputs (RO).

Digital data is transferred from the core storage to the register output features. The contents of each output register are then transmitted to customer-owned devices. The output of this register remains latched until changed by another data transfer. In this also we have data channel adapter like digital input feature which can be operated using external synchronisation. An XIO instruction (initialise write-synchronized) is executed to load the addressed group with the data word. When the data word has been loaded, the external sync 'ready' signal is turned on, signalling the external device that the data is available.

The external device reads the data and then sends an external sync signal to the DO adapter, resetting the external sync 'ready' signal and signalling that the external device is ready for another data word. This initiates a cycle steal request. At the completion of cycle steal operation, the ready signal is turned on. When the last data word of a data table has been loaded, an end of table interrupt is received with DO busy indicator remaining on. The DO busy indicator is on during the last data word until the last external sync signals the adapter that the last word has been loaded.

CHAPTER III

LINE PROTOCOL DESIGN

Line protocols are at the lowest level in the hierarchy of protocols between two computers ready to exchange information and are concerned with details of communication at line level. In the first chapter some of the functions of line protocols were discussed. In this chapter, we describe the line protocols used by this network and how it fulfills those various functions.

The transmission of data between two computers is byte-oriented in which each byte contains eight bits which are transmitted serially one by one onto the communication line. Successive bytes of a message follow each other in a synchronous mode. The data to be transmitted can be of any length and may be divided into a number of messages of variable length. Our interface hardware is so flexible that it can transmit messages of any length. Software takes care of segmenting data into messages and appending sequence numbers to them for transmitting through the hardware to the communication line.

In our line protocol, we have three control characters, which play a vital part in the communication between the two processors. They synchronisation character SYN (16_{16}), start of message character SOM (01_{16}) and acknowledgement character ACK ($7C_{16}$), each of eight bits length. These

three characters are recognised by the hardware in the receiving interface. Let us discuss briefly the importance of each of these characters in the line protocol mechanism.

The first and foremost requirement of line protocols is the synchronisation between two communicating processors [DAVIES, BAR, BER]. Synchronisation must be done at all levels, such as bit level, byte level and message level. In our network bit level and byte level synchronisation are done by the hardware interfaces. While message level synchronisation is done by the software. Bit synchronisation is achieved in the receiving interface automatically by deriving the clock from the serial data received from the line. This is explained in more detail in Chapter 5.

When there is no data to be transmitted from the computer, the sending interface goes on sending an idle 'synchronisation character over the line. After power on, the receiving interface looks at the incoming bit stream and tries to identify the syn character. This character has a special bit combination which can be recognized uniquely in long stream of syns. Once this combination has been achieved,... the interface merely assumes successive eight bits to form a byte.

Message transmission is initiated by the sending processor, by setting a flag in the sending interface. As soon as the flag is set, sending interface goes on getting data from the processor and sends it on to the line at regular intervals. The first byte of the message must be either SOM character or Ack character. If the processor wants to send a message, then the first character will be SOM, otherwise it will be ACK. It should be noted that the sending interface does not generate these characters. When the receiving interface receives SOM character and detects it, it sets a flag and interrupts the processor for handing over the incoming data to the processor.

After sending the message, the sending processor waits for a certain period called time-out, to receive the acknowldgement of the message from the receiving processor. If it does not get the acknowledgement within the time out period, then the sending processor may send the same message again. If the receiving interface gets an acknowledgment character (ACK), it sends an interrupt signal to the processor that it has received the acknowledgement for the last message the processor transmitted. When this happens, the processor disables the timer and may take action to send the next message. An acknowledgement is sent by the receiving processor only if the message received was error-free. No negative acknowledgements are

sent. There is a provision for a one-bit sequence number in the ACK character itself to identify the message for which the acknowledgement is being sent.

Error detection and recovery from errors are an important part of the functions of the receiving interface. The receiving processor must know whether the received message is error-free or not. Most standard protocols use block error detecting codes such as cyclic codes to detect errors in the received message. In our network, line lengths are small and the error rates on the coaxial cables used are expected to be extremely small. Block codes were thought to be inefficient under such conditions and we decided to use a simple parity-checking scheme for each byte of data.

In this scheme, the sending interface appends one parity bit to every byte it transmits on to the line. Here we used odd-parity. The receiving interface checks the parity of every byte it received and sets an error flag if an error is detected. When the receiving processor finishes receiving the messages it checks this flag to find whether the message contained an error.

Implementation of these line-protocols are described in Chapter 4 and Chapter 5 of this thesis.

CHAPTER IV

HARDWARE DESIGN OF SENDING INTERFACE FOR IBM 1800

The hardware interface at IBM 1800 consists of two parts, the sending interface and the receiving interface. The function of the sending interface is to get the 16-bit word from the 1800 and convert it to serial mode and transmit over the line. For every word it receives from the 1800, the SI sends a control signal to get the next word from 1800. This process continues till the message transfer is completed.

The operation of the sending interface may be briefly summarised as follows:

1. In the idle state, the sending interface continuously sends the SYN character over the line.

2. When the PC wants to send a packet, it loads the first word of the packet into the data-register buffer and sets a flag (Mode) in sending interface.

3. Sending interface loads the lower byte from the data register into a shift register (TX RG). This achieves parallel to serial conversion.

4. A 1-bit odd parity is generated and appended to the byte contained in TXRG.

5. The data is transmitted serially from T x RG over the communication line.

6. After the transmission of lower byte, upper byte is loaded into the shift Register (T x RG) from the data register. After sensing the 'ready' signal from the 1800, the SI sends a 'sync' pulse to the 1800, for the next word.

7. The data channel loads the next word of the packet into the data register, and process continues at Step 3.

8. When all the words are transmitted, the PC sets the Mode flag in the SI off and the interface goes back to the idle state in Step 1.

All the above steps are explained in the hardware description language [CHU].

4-1. HARDWARE DESCRIPTION OF SENDING INTERFACE

/ $\overline{\text{Mode}}$. (LC=0) . PP/ T x RG ← 'sync' character, BF ← 0

/Mode . $\overline{\text{BF}}$. (LC = 0) . PP/ T x RG ← lower byte of data reg, BF ← 1

/Mode . BF . (LC = 0) . PP/ T x RG ← upper byte of data reg, BF ← 0

/Mode . BF . (LC = 0) . Ready . PP /
Send 'sync Pulse' to the 1800

/ (LC ≠ 0) . PP/ Transmit T x RG on line,
Shift Right T x RG,
Decrement line counter

/ (LC = 0) . PP/ LC ← 9

Registers: TXRG (0-7)
Data Register (0-15)

Clock PP

Counter Line counter LC : Counter of 9

Flags Mode, BF

Mode is set and reset by the processor

When Mode = 0 then 'SYN' character is transmitted over the line

Mode = 1 then data (either lower byte or upper byte) is transmitted depending on byte flag (BF)

when BF = 0, lower order byte is transmitted

BF = 1, upper order byte is transmitted.

All the above operations are parallel and happen when the conditions given in the slashes are true. We now describe how it is implemented in hardware.

4-2. BASIC BUILDING BLOCKS OF SENDING INTERFACE

The sending interface is divided into four sub blocks

1. system clock
2. transmission unit
3. control block
4. encoding and driver unit.

These blocks are interconnected as shown in Figure 4-1.

4-3. SYSTEM CLOCK

As the chosen line capacity is 10 KB/Sec. a 10KHz clock is used for driving the transmission register and the line counter. This 10KHz clock is obtained from a 2MHz crystal oscillator after dividing successively by five, ten and four. For getting this 2MHz frequency clock, a well stabilised crystal clock generator is made use of with high power supply rejection. This 10KHz clock is our system clock (PP) in the sending interface. A simplified schematic is shown in Figure 4-2; the detailed diagram is given in Appendix A. This clock drives the line counter and the transmission register.

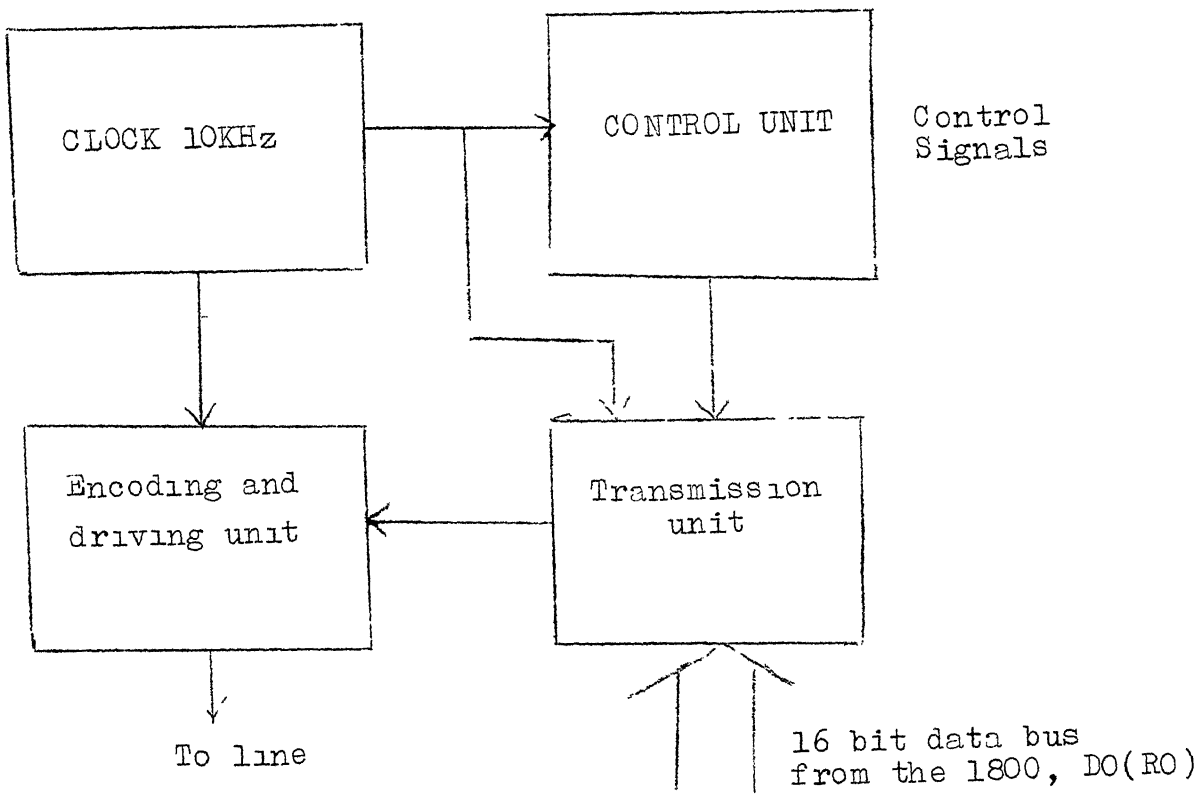


Figure 4-1: Block Diagram of SI

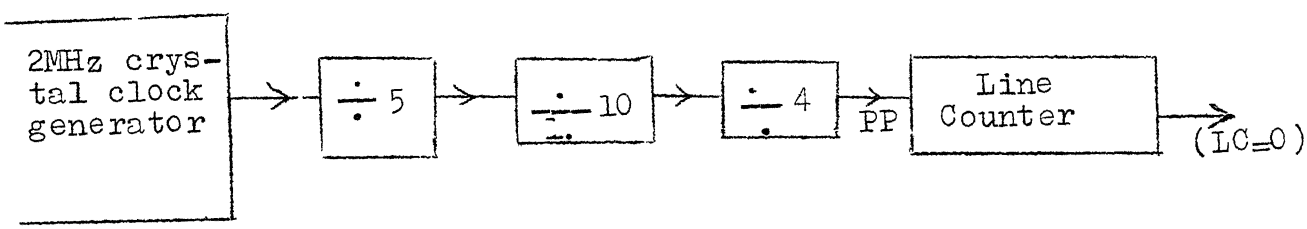


Figure 4-2: System Clock

4-4. TRANSMISSION UNIT

The transmission unit consists of 2 eight bit buffers, 8 four-line to one-line multiplexors and a transmission register. Digital output (DD-A0) from the IBM 1800 loads the 2 eight bit buffers with the data from its memory. The outputs from the buffers and the 'synchronisation character' are input to the multiplexors. The multiplexors select either SYN character or lower byte of data or upper byte of data depending on the two control signals given to the multiplexors. The output from the multiplexors which are 8 bits directly go to the transmission register which has a length of 8 bits and which takes data parallely and transfers it serially when shift pulses are applied to it. The data loaded in the transmission register is also sent in parallel to the parity generator which generates an odd parity bit. This bit is connected to the serial input of the transmission register. So the parity gets shifted along with the data and is transmitted over the line as the ninth bit of byte. The schematic of this is shown in Figure 4-3, a more detailed diagram is given in Appendix A.

4-5. CONTROL UNIT

The control unit is the heart of the system from where all the control signals are sent to appropriate places at required times. There are two flags in the system that are responsible for the control signals. The first flag is a

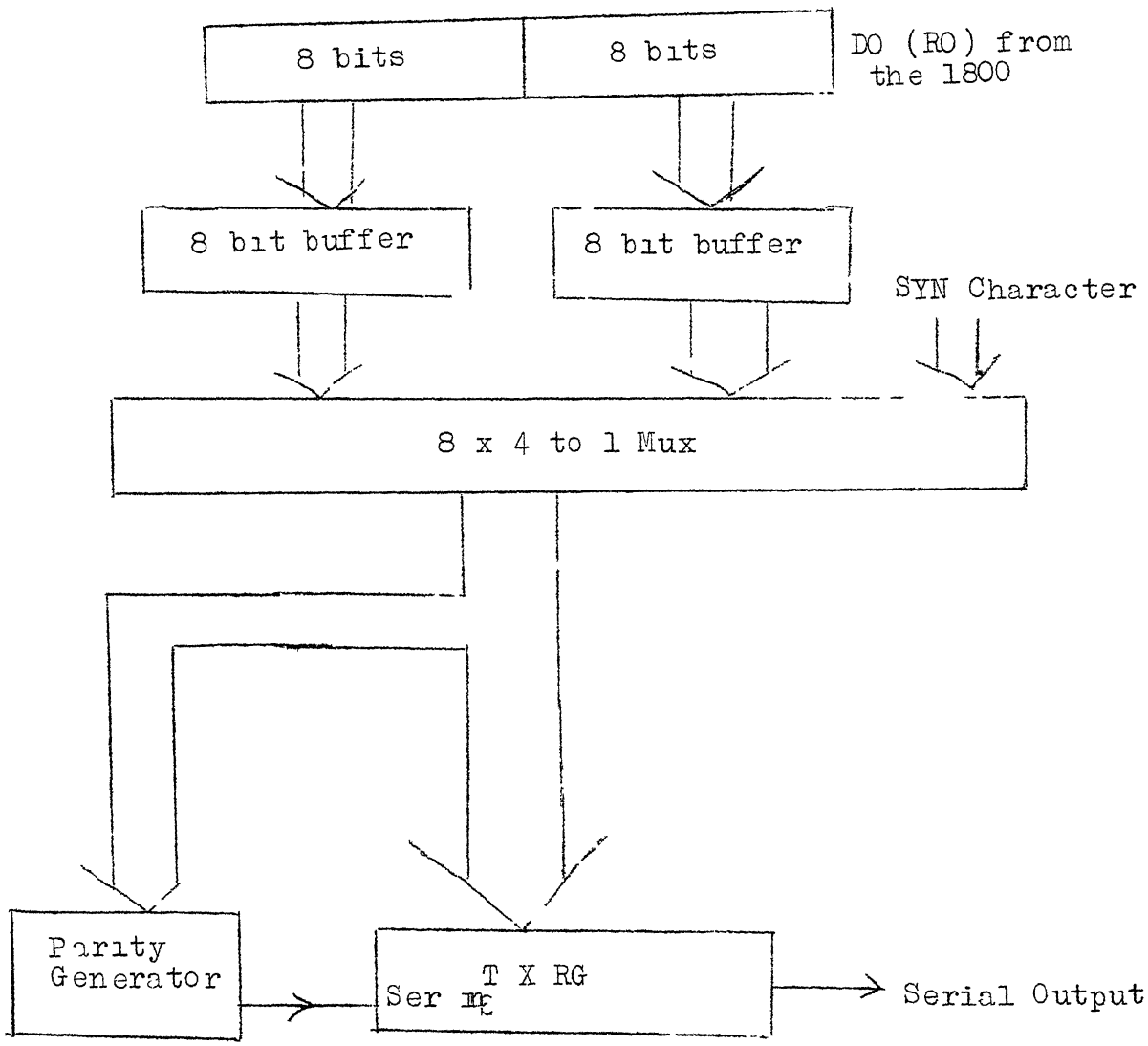


Figure 4-3: Transmission Unit.

mode flag which is set by the processor before sending a message and reset by it after sending is complete. As the 1800 data channel transfers one word from memory at a time, there must be some mechanism in the interface to distinguish whether the byte to be transmitted is the lower or upper byte of the word. This is indicated by a flag BF known as byte flag and it goes on toggling for each byte transmission.

The 1800 sends data to the interface through data channel operation with external 'sync' option. For every word it gives to the interface, it sends a 'Ready' signal to the interface. When the interface transmits the lower byte of data and is loading the upper byte of data in the transmission register, it sends a 'synchronisation pulse' to the 1800 data channel, indicating that the interface is ready to receive the next word and transmit it. This process goes on till the end of message transfer. So the interface's control unit must sense the 'ready' signal and send the 'sync' pulse to 1800 at appropriate intervals.

The multiplexor also needs control signals for its operation. The multiplexor input has three buses (SYNC character, lower byte of data and upper byte of data), the multiplexor has to select one of the three buses and route it to the transmission register for loading. So for selecting one out of three lines, we need two control signals provided by a mode and byte flags. Timings of these signals is given in Figure 4-4. Detailed hardware description of control

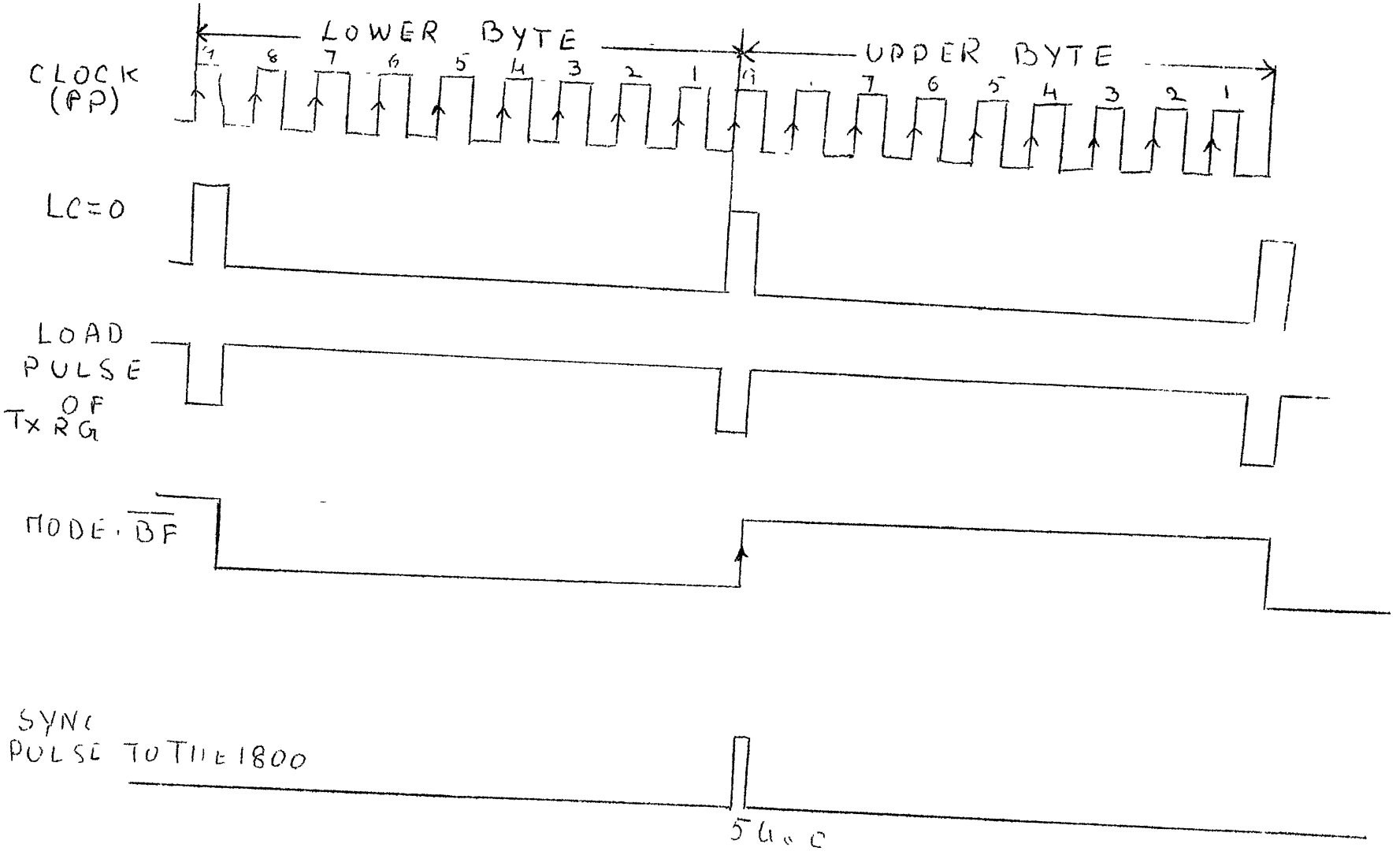


FIG 4.4 TIMING DIAGRAMS.

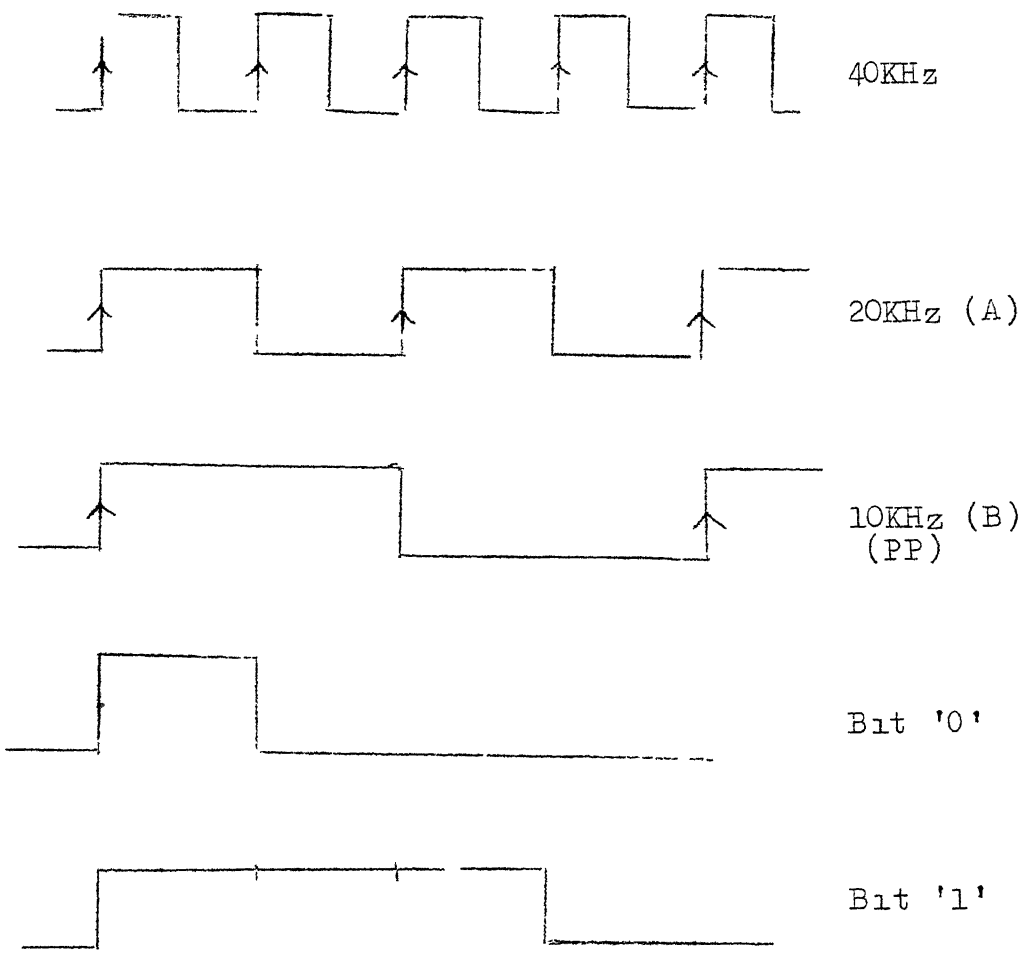


Figure 4-5: Encoding Diagrams.

unit is given in Appendix A.

4-6. ENCODER AND DRIVER UNIT

The serial output of the T x RG is coded to get proper transitions at the start of each bit. With these transitions, the clock of the receiving interface is synchronised. The coding used is shown in Figure 4-5. It can be seen that the signal transmitted for bit 0 remains 1 for the first 0.25 of the basic clock (PP) and is 0 for the remaining 0.75 part. The signal for bit 1 is 1 for the first 0.75 part of PP and is 0 for the remaining 0.25 part. The logic for coding can be easily derived as

Serial Encoded output = I.A + I.B + A.B

where A is the 20KHz clock, B is the 10KHz clock and I is the serial output of the shift register.

The encoded output is fed to the line driver. This line driver consists of two transistors capable of driving upto 200 feet without any distortion. A more detailed circuit diagram of the encoder and line driver unit is given in Appendix A.

CHAPTER V

HARDWARE DESIGN OF RECEIVING INTERFACE FOR IBM 1800

The hardware design and implementation of the receiving interface is discussed in detail in this chapter. The main function of the receiving interface is to get synchronized with the incoming data, convert the serial data to a parallel form and transfer it to the computer.

The operation of the receiving interface may be briefly summarised as follows:

1. In the normal mode, the receiving interface continuously receives the SYN character from the line. The RI derives the clock from the serial data it receives and sets 'synchronisation flag (SF) in the interface setting of the SF indicates that the byte synchronisation has been achieved.

2. The serial data is connected to the serial input of a shift register (R x RG) and the derived clock is used as shift pulses of the shift register. This achieves serial to parallel conversion. The outputs from the R x RG are connected to 2 eight bit buffers from where the 16-bit data goes to the 1800.

3. If the RI detects an ACK character then it loads the character into the lower order buffer and sends an interrupt signal to the 1800 to take the ACK character into the memory.

If the RI detects an SOM character then it loads the character into the lower order buffer, sets a mode flag and interrupts the processor to initiate the data channel operation for reading the inputs. The data channel sends a 'ready signal' to the RI.

4. After filling the upper order buffer with the next byte the RI sends an 'external sync' pulse to the 1800 data channel to read the 16-bit word. After reading the word, the data channel sends a 'ready' signal. This process continues till the end of message transfer.

5. When all the words are received, the processor resets the mode flag in RI and the interface goes back to the normal state in Step 1.

6. During the message reception if any parity error occurs, then the RI sets a parity error flag that can be accessed by the processor.

5-1. HARDWARE DESCRIPTION OF THE RECEIVING INTERFACE

$/\overline{SF}$. PP/ If (R x RG = Syn character) then SF $\leftarrow$ ' ',
LC $\leftarrow$ 9, Mode $\leftarrow$ 0, BF $\leftarrow$ 0

/PP/ Shift Right R x RG
R x RG(8) $\leftarrow$ bit from the communication line

/(LC $\neq$ 0) . PP/ decrement line counter

/(LC = 0) . PP/ LC $\leftarrow$ 9

/SF . $\overline{\text{Mode}}$. (LC = 0) . $\overline{\text{Parity}}$. PP/ SF $\leftarrow$ 0

/SF . $\overline{\text{Mode}}$. (LC = 0) . $\overline{\text{Parity}}$. PP/ If R x RG = ACK character
then lower byte (Data-Reg) $\leftarrow$ R x RG,
Interrupt Processor

If R x RG = SOM character

then lower byte (Data-Reg*) ← R x RG,

Interrupt PC, Mode ← 1, BF ← 1

If R x RG ≠ SYN then SF ← 0

/SF . Mode . (LC = 0) . BF . PP/ upper byte (Data-Reg) ← R x RG, BF ← 0

/SF . Mode . (LC = 0) . $\overline{BF}$. PP/ lower byte (Data-Reg) ← R x RG, BF ← 1

/SF . Mode . (LC = 7) . $\overline{BF}$. 'Ready Signal' . PP/ Send synchronisation pulse to the 1800.

/Mode . $\overline{Parity}$. (LC = 0) . PP/ set parity error flag.

Registers	Data-Register (0-15) Receiving Register R x RG (0-7)
Counter	line counter LC : Counter of 9
block	PP
Flags	Synchronisation flag (SF) Mode flag Byte flag (BF) Parity error flag.

When the conditions in the slashes become true, then the corresponding commands are executed. At each clock pulse one or more conditions may become true thereby executing more than one command. The following sections describe how the design expressed in the above language is implemented in hardware.

5-2. BASIC BUILDING BLOCKS OF THE RECEIVING INTERFACE

The receiving interface is made up of four basic building blocks:

1. Bit synchronisation block
2. Data receiving block
3. Byte synchronisation and control unit
4. Level translator card.

A simple schematic diagram of the receiving interface is shown in Figure 5-1.

5-3. BIT SYNCHRONISATION UNIT

The bit synchronisation unit derives the clock from the serial data over the synchronous communication line. This derived clock is used to strobe the data into the serial to parallel converter. The encoded data from the line goes to a +phase locked loop and voltage controlled oscillator unit'. The frequency of the VCO is adjusted to be exactly twice the frequency of the data rate. When the PLL is locked, the clock in the receiving interface is obtained by dividing the VCO frequency. A simple schematic is shown in Figure 5-2.

5-4. DATA RECEIVING BLOCK

The derived clock is used to shift a serial to parallel converter or shift register and the serial data is taken as serial input to the shift register of eight bits length. The eight outputs from the shift register are input to 2 eight bit buffers where the data is strobed into the buffers at

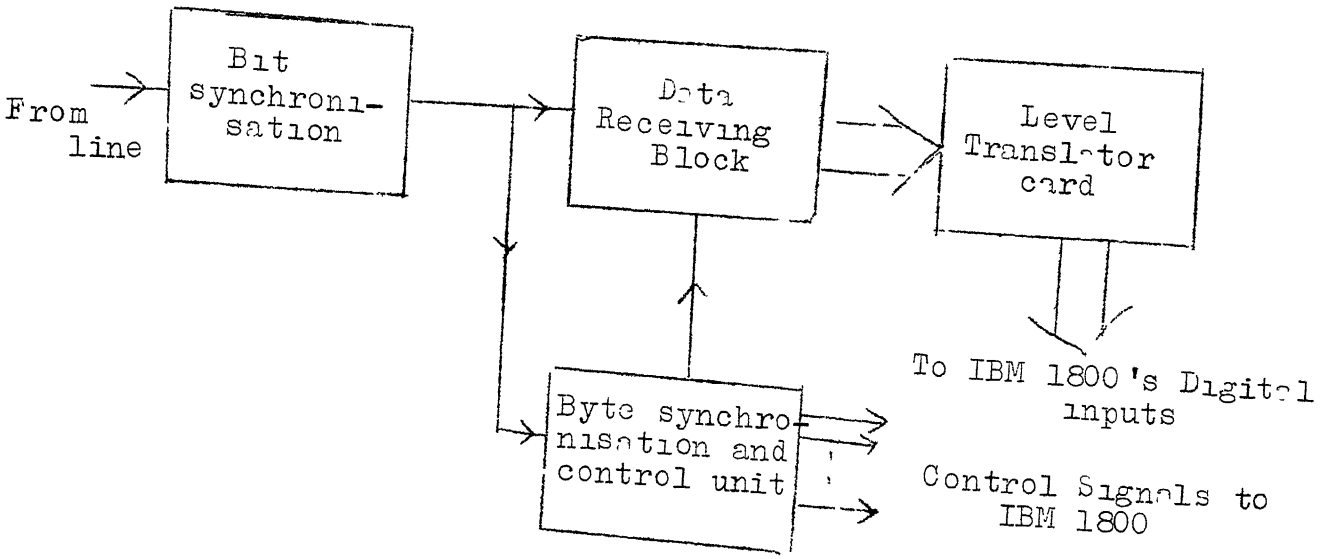


Figure 5-1: Block Diagram of RI.

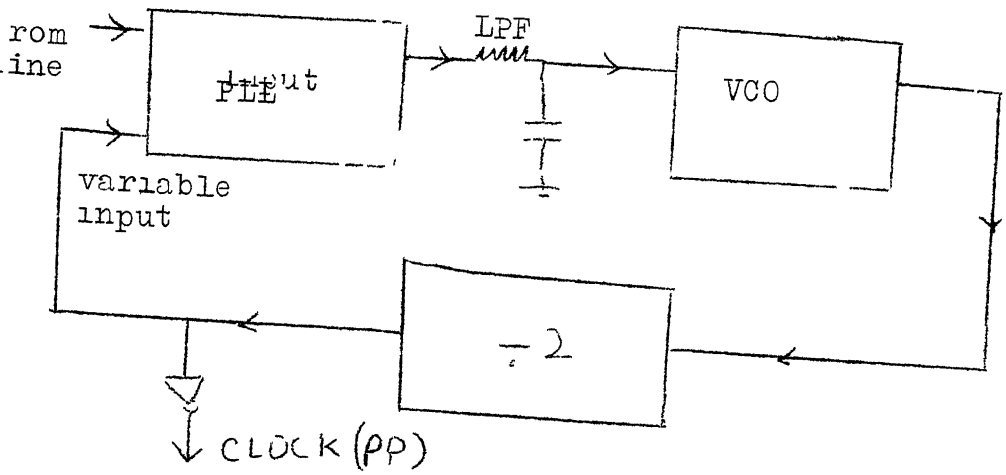


Figure 5-2: Bit Synchronization Unit.

appropriate time intervals. In this unit basically the serial data is converted into parallel form and gives as inputs to buffers. A simplified diagram is given in Figure 5-3.

5-5. BYTE SYNCHRONISATION AND CONTROL UNIT

Byte synchronisation is achieved by setting the synchronised flag (SF), when the first SYN character is detected by the interface. At the same time the line counter is loaded with nine and it goes on decrementing with each clock pulse. SF is reset only when parity error occurs in the data or some other character is received other than SOM or ACK by the interface. When SOM character is received by the interface it sets a 'Mode flag' and at the same time sends an interrupt signal to IBM 1800. As long as the mode is set, the interface goes on receiving data and converts it to parallel form and loads the buffers. In the interrupt routine for SOM, the 1800 initiates the data channel operation to read the data with external sync pulse.

The 1800 sends a ready signal to the interface and after loading the buffers, interface sends a synchronisation pulse of 5μsec duration to the 1800 to read the 16-bit word from the buffers. This process goes on till the end of message transfer to the 1800. Mode flag is reset by the 1800 after receiving the message completely. If an ACK

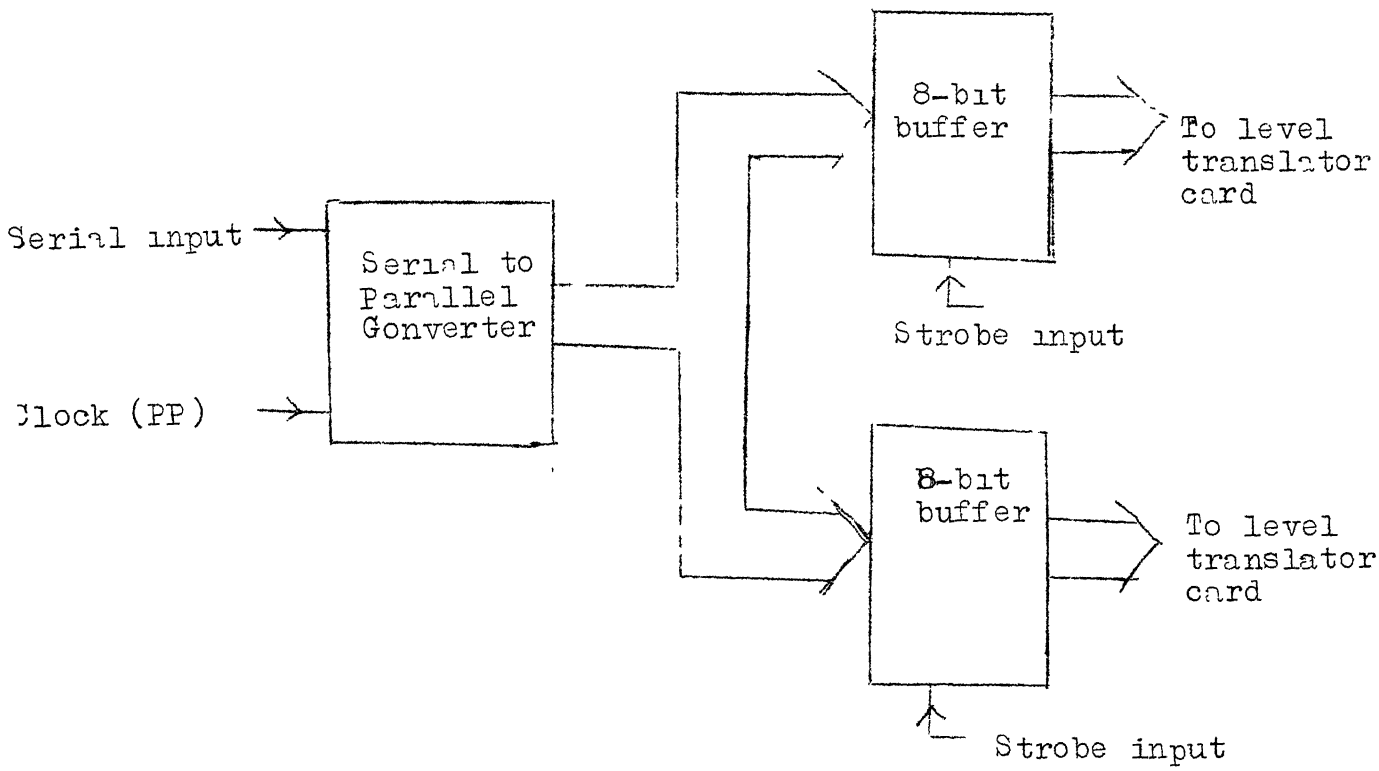


Figure 5-3: Reception Unit.

character is received by the interface when the mode flag is reset, the interface sends an interrupt signal to the processor. Setting of Mode Flag indicates that the interface is in message mode and receives any combination of bits. Thus the transparency of code has been achieved with the mode flag.

For selecting the buffers, there is another flag known as byte flag (BF) which goes on toggling after loading each buffer. This flag is clocked by the loading pulse of the buffer. When a parity error occurs in the received data in the receiving mode, then it will set a flag, which can be accessed by the 1800 to check the parity of the message, known as parity error flag. So the control unit contains the circuitry for setting or resetting of the above flags and is shown in detail in the Appendix B.

5-6. LEVEL TRANSLATOR CARD

This card contains the circuitry to convert the TTL-voltage levels to IBM 1800 voltage levels. The sixteen outputs from the TTL buffers, SOM interrupt signal, ACK interrupt signal and 'sync' pulse to the 1800 are inputs to the level translator card. The sixteen outputs go to the Digital Input (DI) group of IBM 1800, from where they are read into the memory. The SOM interrupt signal and ACK interrupt signal are terminated on PISW (Process Interrupt Status Word) for interrupting the processor, 'Ready signal'

from the 1800 is converted to a TTL-voltage level in this card and is given as input to the interface.

The complete logic diagrams are given in Appendix B and should be consulted for a thorough understanding of the control circuitry.

-

CHAPTER VI

CONCLUSION

The main aim of the project was to link up TDC-316, IBM 1800 and DEC-1090 in a star network with MICRO-78 as the switching centre. Work on the MICRO-78 interface to the DEC-1090 could be started only very recently since DECNET protocols arrived late. Also, the MICRO-78 which was supposed to be at the nodal point has not yet arrived. So initial testing of all the three interfaces for IBM 1800, TDC-316 and MICRO-78 was done by connecting the sending interface of the machines to their own receiving interfaces. This loop-back testing was successful.

As TDC-316 and IBM 1800 interfaces were already operational in the Computer Centre, they were directly connected together and messages transferred between them. All the combinations from 0 to FF/16 were generated, transmitted and received without any error. Any further work can be done only after MICRO-78 arrives and DEC-1090 hardware is ready. So we can hope the network will be completely functional by July 1980.

After analysing the network critically, the following points came out of the various aspects of the system.

1. From the protocols, it could be seen that ACK and SOM are treated identically by the PC. Both interrupt the PC and further action is taken by the PC after identifying ACK or SOM. Immediately the question arises whether it was necessary to have an ACK character all.

Why not recognise only SOM and decide to send acknowledgement as any other message? Thus, recognition of ACK by hardware is a redundant feature.

2. There is no flexibility as far as data rate is concerned. Some hardware modifications will be necessary if data rate is to be changed. In the receiving interface the free running frequency of the voltage controlled oscillator has to be changed for a different data rate.

Further work on this work can proceed in two directions:

[a] The hardware development could be done by implementing a DMA interface for MICRO-78 and TDC-316. With the DMA interface implementation, it should be possible to go to much higher rates.

[b] It should be possible to modify the present hardware to form a data loop. This would be complicated by the fact that there is provision for two synchronous lines on the DLC-10 side. It would need a sort of direct linking between the sending interface of any PC to its own receiving interface and also decoding circuits which could identify the source and the destination of the packet received.

A lot of work can be done on the software part. The software work could start with the design of a network operating system, alongwith a network language. Also performance studies could be conducted on the networks, e.g., effect of packet length on error rates and processor utilisation studies etc.

This network can be further extended by connecting a separate MICRO-78 at each nodal computer than can serve as a communication processor. The nodal computer communicates with its communication processor which in turn can communicate with the central switch. By this method, we have two more nodal points at each communication processor which can be connected to other computers or MICRO-78s for the enlargement of the network.

-

REFERENCES

[ABRA and KUO] Abramson, N., and Kuo, F.F. (eds) : Computer Communications networks (1973).

[BRAN 72] Brandit, G.J., and Chretien, G.J., Methods to control and operate a message switching network in proceedings of the symposium on Computer Communications Networks and Teletraphic (1972), 263-276.

[CHU] Chu: Computer organisation and microprogramming.

[CROW 75] Crowther, W.R., et al: Issues in packet switching network design, Proc. AFIPS NCC 44 (1975), 161-175.

[DAVI 68] Davies, D.W., The principles of a data communication network for computers and peripherals, in Advances in Computer Communications (1974) edited by Chu, W.W.

[DAVIES and BARBER] Communication networks for computers (1973).

[DDAS 79] Debashish Das, IIT/K Computer Network - MICRO-78 Hardware Interface.

[FRAN 72] Frank, H., and Chow, W, Topological Optimization of computer networks : Proc. IEEE 60, 11(Nov. 1972), 1385-1395.

[FRAS 76] Fraser, A.G., The present status and future trends in computer/communication technology : Computer (Sept. 1976), 10-19.

[FUN 1800] Functional characteristics of IBM 1800.

[GRAY 72] Gray, J.P., Line control procedures : Proc. IEEE, 60,11 (Nov. 1972), 1301-1312.

[KAHN 72] Kahn, R.E., Resource sharing computer communications networks, Proc. IEEE 60, 11 (Nov. 1972), 1397-1407.

[KLE I] Kleinrock, L., Queuing systems, Volume II, Computer Applications (1976).

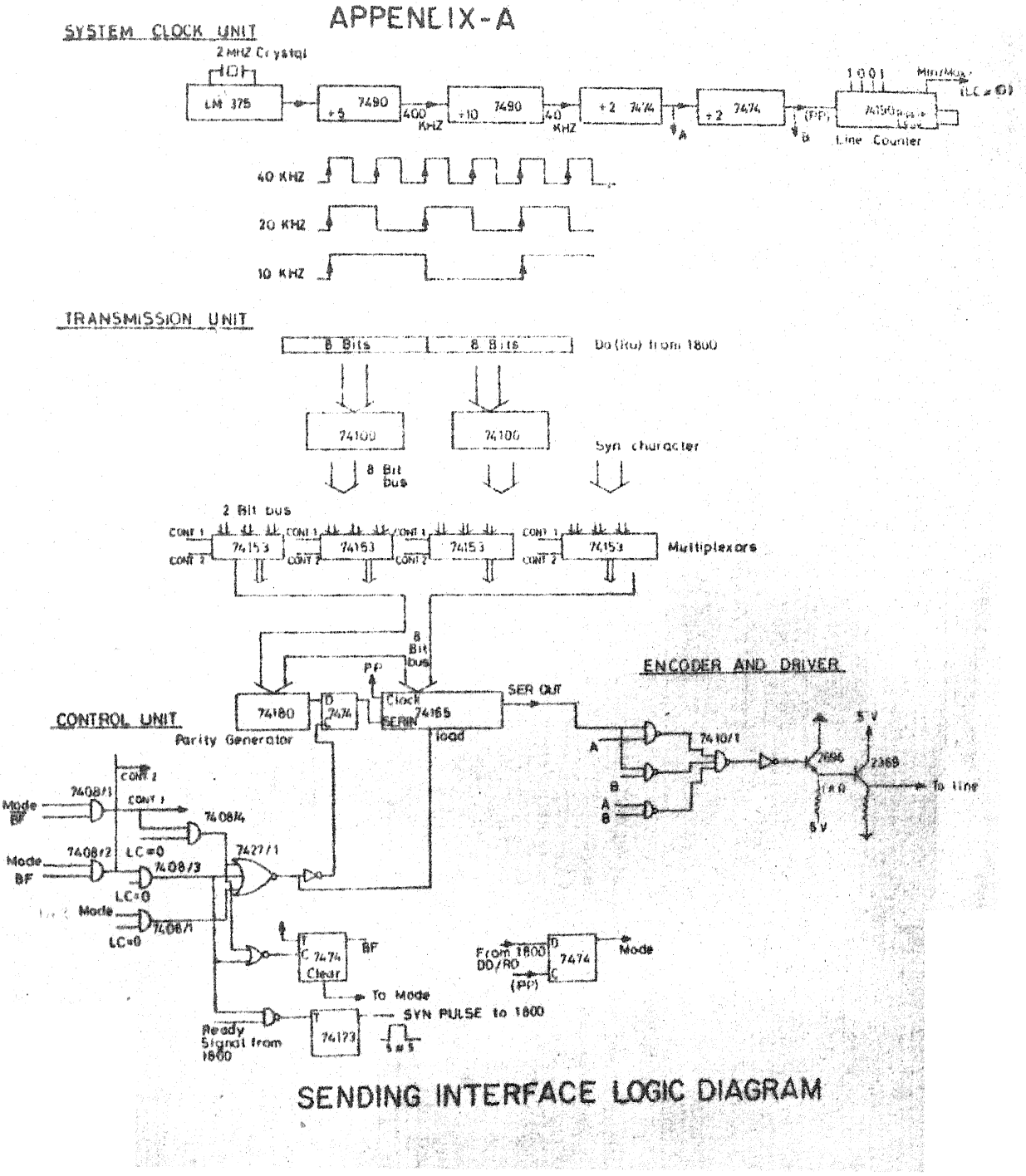
[NSN 79] Narayanan, N.S., IIT/K Computer Network - TDC-316 Hardware Interface.

[ORNS 72] Ornstein, S.M., et al, The terminal IMP for the ARPA Computer Network : Proc. AFIPS SJCC 40(1972), 243-254.

[STUT 72] Stutzman, A.B.W., Data Communication Control Procedures : Computing Surveys, 4,4(Dec. 1972),197-220.

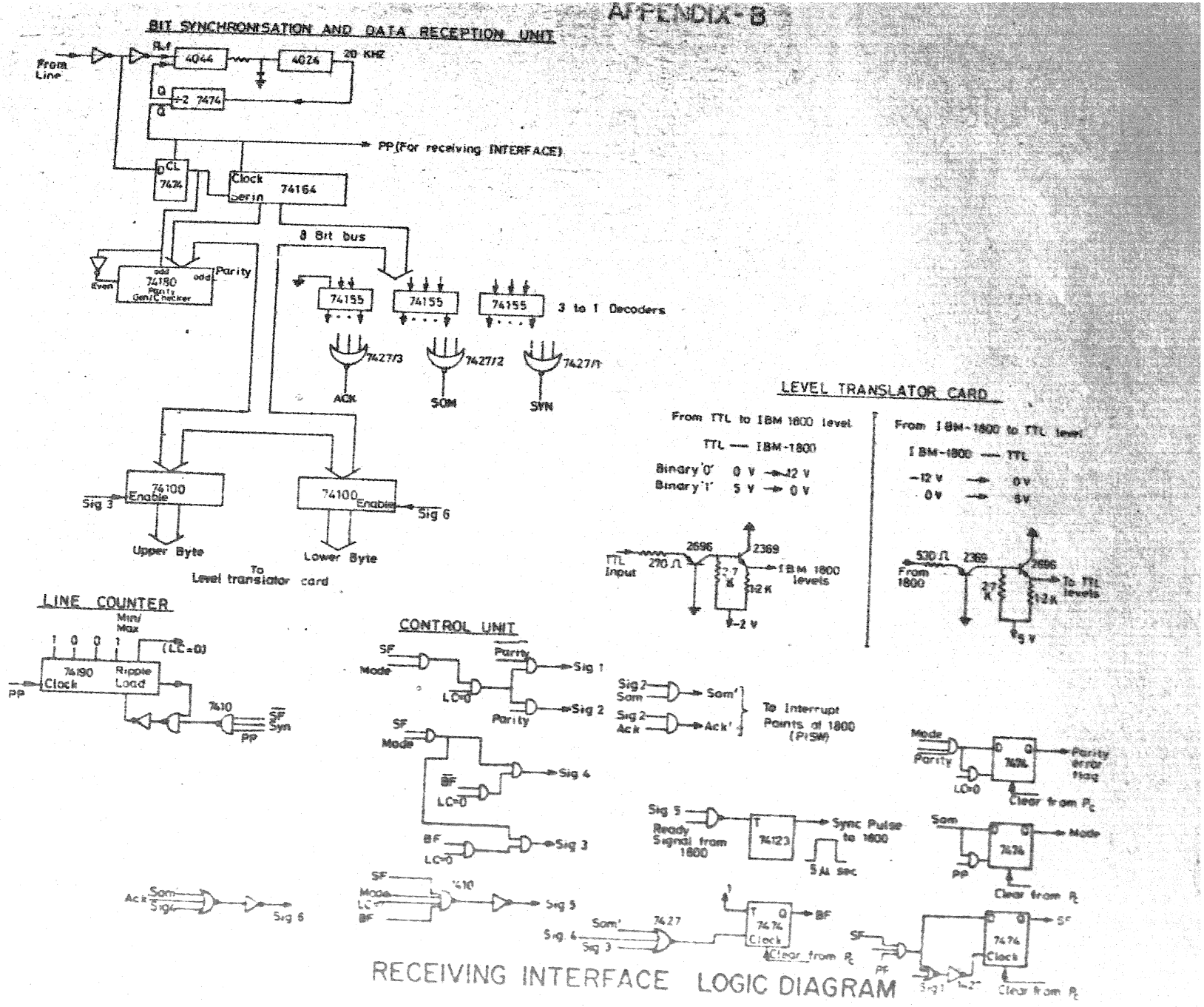
[WALD 75], Walden, D.C., Experience in building, opeeating, and using the ARPA network : 2nd USA-Japan Comp. Conf., Tokyo (Aug. 1975).

APPENDIX-A



SENDING INTERFACE LOGIC DIAGRAM

APPENDIX-B



RECEIVING INTERFACE LOGIC DIAGRAM